DOCUMENT RESUME

ED 308 835 IR 013 895

AUTHOR Ross, Steven M.; McCormick, Deborah
TITLE Computer Access and Flowcharting as Variables in Learning Computer Programming.
PUB DATE Feb 89
NOTE 10p.; In: Proceedings of Selected Research Papers presented at the Annual Meeting of the Association for Educational Communications and Technology (Dallas, TX, February 1-5, 1989). For the complete proceedings, see IR 013 865.
PUB TYPE Reports - Research/Technical (143) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Academic Achievement; *Cognitive Style; *Computer Literacy; *Flow Charts; Learning Strategies; Likert Scales; Media Research; Multivariate Analysis; *Programing; Secondary Education; Sex Differences; *Student Attitudes
IDENTIFIERS BASIC Programing Language

ABSTRACT

Manipulation of flowcharting was crossed with in-class computer access to examine flowcharting effects in the traditional lecture/laboratory setting and in a classroom setting where online time was replaced with manual simulation. Seventy-two high school students (24 male and 48 female) enrolled in a computer literacy course served as subjects. None had received any programming instruction. Four treatment groups were arranged by crossing the computer access variable (unlimited vs. limited) with flowcharting (required vs. not required), and five instructional units dealing with introductory concepts in the BASIC programming language were presented in eighteen 50-minute class periods. Results showed that group means on four of five posttest measures were directionally higher for the limited-access group than for the unlimited-access group; females performed better under limited access, while males showed the opposite pattern; and low-ability students performed better with unlimited access, while middle- and high-ability students performed conversely. In addition, students performed better when not required to submit flowcharts, and flowcharting appeared to be regarded by many students as an entirely separate task rather than as a programming aid. Gender was not significantly related to either achievement or attitude. Three important considerations for the teaching of programming are suggested by these results: (1) unlimited computer access may be less important than is generally assumed; (2) more time should be given to instruction in flowcharting than is typically done; and (3) males and females appear to have the same potential and instructional needs for learning programming. (12 references) (CGD)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED308835

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☒ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Title:

Computer Access and Flowcharting as Variables in Learning Computer Programming

Authors:

Steven M. Ross
Deborah McCormick

TR013895

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Michael Simonson

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Computer Access and Flowcharting as Variables in Learning Computer Programming

Steven M. Ross and Deborah McCormick

Foundations of Education
Memphis State University
Memphis, TN 38152

Computer Access and Flowcharting as Variables
in Learning Computer Programming

Research on pre-college programming instruction has often yielded disappointing results. After completing such courses, students frequently exhibit fundamental misconceptions and poor problem-solving strategies (Dalbey, Tournaire, & Linn, 1986). These outcomes suggest that traditional methods of teaching programming in secondary schools may need to be reoriented to better accommodate environments that often differ substantially from college classes. One interest of the present study concerns the provision of unlimited access to computers in programming classes. While the importance of computer access is intuitively apparent, the notion has not been thoroughly researched. On the one hand, it would seem that on-line practice can promote discovery of design principles and procedures (Bayman & Mayer, 1983). On the other hand, the novice's "rush to the computer" may be associated with little program planning and consequently poor techniques and conceptual understanding (Mayer, 1979; Pea & Kurland, 1983; Dalby et al., 1986). From the latter perspective, limited computer access might engage students to a greater extent in planning activities, and thus engender more systematic problem-solving approaches than would occur with unlimited access.

The above rationale raises the additional question of whether planning activities can be facilitated by using flowcharts to specify the program logic and structure prior to writing code. Research on the benefits of flowcharting, however, has been inconclusive (Glorfeld & Palko, 1984; Schneiderman et al., 1982; Brook & Duncan, 1982). Accordingly, as an extension of this earlier work, the present study crossed the manipulation of flowcharting with in-class computer access, thus examining flowcharting effects in (a) the traditional lecture/laboratory setting and (b) a classroom setting replacing on-line time with manual simulation. A reasonable assumption is that when computer access is not an immediate option, students will use flowcharting more effectively as a design strategy. A supplementary question concerned whether gender differences in programming performance would occur under the different flowcharting and computer access conditions. Previous results suggest that females generally perform as well as males at learning to program (Madinach & Corno, 1985; and numerous others), but it could be that different attitudes and learning strategies (Lockheed, 1985) predispose one group to adapt better in less structured (e.g., no flow diagram; unlimited access) settings, and vice versa.

Method

Sample and Design

Subjects were 24 male and 48 female high school students enrolled in four sections of a computer literacy course taught by the second author. All had completed 12 weeks of the 18-week course, but had not yet received any programming instruction. They were assigned at random to four treatment groups arranged by crossing the computer access variable (unlimited vs. limited) with flowcharting (required vs. not required).

Initial analysis used a 2 (computer access) x 2 (flowcharting) x 2 (gender) MANOVA with dependent variables consisting of error recognition, interpretation, programming templates, mental models, programming problem,

flowchart score, programming attitude, and flowcharting attitude. Subsequent analysis, incorporating prior academic achievement as a variable, consisted of a 2 x 2 x 3 MANOVA utilizing computer access, flowcharting, and ability group (high, medium, or low) as independent variables.

Course Curriculum and Materials

Five instructional units dealing with introductory concepts in the BASIC programming language were presented in 18 50-minute class periods. Unit I dealt with operational commands, such as RUN, LIST, SAVE, and editing features. Students were also introduced to programming terminology and fundamental concepts, as well as flowcharting techniques. Units II - V presented applications using programming statements LET, PRINT, GOTO, INPUT, and IF/THEN. The posttests described below were administered over a two-day period immediately following the completion of Unit V.

Achievement posttest. The achievement posttest assessed performance on five fundamental programming skills and on ability to utilize flowcharting as a programming aid.

Subtest I, "Error Recognition," presented eight short groups of programming code which contained either a syntax or logic error. Students were asked to find and specify the cause of the error. Items were scored as correct (1) or incorrect (0).

Subtest II, "Interpretation," presented students with four short programs and required them to identify the purpose and output of the code. Students were scored for both their narrative explanations and descriptions of output (1 for correct, 0 for incorrect).

Subtest III, "Programming Templates" required students to write three short segments of programming code which performed a fundamental procedure such as averaging, counting, or evaluating input. Programs were scored as correct in both syntax and logic (2), correct in either logic or syntax (1), or totally incorrect (0).

Subtest IV, "Mental Models," was designed from the work of Bayman (1983) to assess accuracy of students' mental models of statement execution. Students were given common programming statements and asked to describe in a step-by-step fashion exactly what happened when the statement executed. Explanations were scored as correct (2), incomplete (1), or incorrect (0).

Subtest V, "Programming Problem," presented students with a problem for which they had to develop a program. Students first designed a flowchart and then wrote programming code. Flowcharts were scored as correct (3), incomplete (2), incorrect (1), or no attempt (0). Programming score was obtained by scoring individually five routines or templates required by the programming problem (input routine; evaluation of input; use of summing variable; combining a literal string and a variable in a print statement; and recursion) in the same manner as Part III above (2, 1, or 0).

In scoring the posttest, clear rules were developed and strictly followed to ensure objectivity. Also, identification numbers, rather than names, were used in order to conceal subjects' identities during scoring. Internal consistency alpha's ranged from .65 to .86 for all measures.

BEST COPY AVAILABLE

Attitude measure. Following the posttest, a 15-item attitude survey was administered. Students were asked to respond to such statements as "Programming is fun" by indicating the extent to which (on a Likert scale of 1 to 5) they agreed with the item. Statements related to either attitude toward use of flowcharts or attitude toward programming. Ten additional items were included to gather information on students' prior programming experience, out of class computer access, and so on.

Procedure

Students in these classes had been accustomed to sharing a computer with one or more partners during the prior 12 weeks of the semester. They were told that for this programming unit, a new method of conducting the class was to be tried, and that on-line computer time and classwork requirements would be different depending on the instructional group in which they were placed. The classes were quite willing to cooperate and, after the first few days of instruction, there was never a need to discuss the different methods again.

Treatment began on Unit II, with the presentation of new concepts and statements to the combined treatment groups. After the instructional period, students practiced the statements that had been presented according to assigned treatment procedures. Students in the unlimited-access group worked in the laboratory section of the classroom where a computer was assigned to each individual to use as he or she desired. Members of the limited-access group worked at their desks in the lecture area of the classroom. These students (ranging in number from 7 - 10) shared a single computer for testing their code. Working with partners or in informal groups was not allowed, although both groups were allowed to discuss problems and seek help from other students or the teacher. It was expected that the limited-access group would be more responsive to interaction with the teacher than the unlimited-access group and, consequently, might receive more instruction and attention. To control for this, the teacher did not initiate contact with either group and responded when called on by asking a leading question or making a suggestion rather than supplying the answer.

Half of the students in each of the access groups were required to submit flowcharts with their assignments. The classes were introduced to the technique of flowcharting as an instructional design aid in the first programming unit. Flowcharting was initially demonstrated in a non-programming application and applied to programming when the first statement was introduced. One class period was devoted to instruction in flowcharting and all students participated in exercises involving flowchart creation. The teacher continued to demonstrate program flow using flowcharts and to recommend them for program design. However, only those students in the flowchart groups were required to submit them to fulfill their assignments. The same basic procedures were followed for Units III-IV. Following Unit V, students completed the achievement posttest followed by the attitude survey.

Results

Dependent variables were scores on the five programming achievement subtests, flowcharting score, flowcharting attitude, and programming attitude. All measures, with the exception of programming attitude, were significantly interrelated (Median r = .55), implying the use of multivariate analysis of variance (MANOVA) to decrease the risk of a Type I error. The basic MANOVA was a three-way factorial consisting of 2(computer access) x 2(flowcharting) x either 2(gender) or 3(ability) group. A regression solution was used in all multivariate and univariate two-factor analyses to control possible biases caused by unequal n's. Initial analyses showed no differences on cumulative attitude score due to either the computer access or flowcharting variable.

Gender as an Individual Difference Variable

Analyses using gender as a grouping variable yielded only one significant effect, the interaction between computer access and gender ($p < .02$). Univariate tests of the interaction, however, were significant only on mental models ($p < .03$): females performed better on mental models in the limited access group (M = 53% correct) than in the unlimited-access group (M = 34%); no differences were evidenced for males. In general, males scored slightly but not significantly ($p > .05$) higher than females on the various programming subtests.

Ability as an Individual Difference Variable

To examine the ATI involving academic ability, students were ranked according to cumulative grade point averages and were divided into high, medium, and low groups of approximately equal numbers based on those rankings. As would be expected, the ability group main effect was significant in the MANOVA ($p < .001$) and in all univariate tests (high > middle > low). More revealing was the significant multivariate computer-access by ability ATI ($p < .01$), which was also significant in all univariate tests except for programming templates. The consistent pattern was for the low-ability group to perform better under unlimited- than limited-computer access, whereas the opposite tendency occurred for middle- and high-ability groups. Follow-up examination of the five ATIs showed that each pattern was disordinal. Specifically, in each case, the unlimited access mean was higher than the limited access mean for low-achievers ($p < .05$), while the converse pattern occurred for middle- and high-achievers.

The MANOVA also yielded a significant flowcharting main effect ($p < .05$). Univariate tests were significant for mental models and programming problem. However, an examination of group means indicated that the effect was not in the direction hypothesized. Students who were not required to submit flowcharts tended to score higher on all subtests than those required to submit them. On mental models, the no-flowchart group mean was 1.13 compared to the flowchart-required group mean of .80; on the programming problem, the respective means were 1.08 and .65.

Discussion

The present results were inconsistent with some common assumptions regarding the influences on learning programming of unlimited in-class

computer access, flowcharting, and gender. Accordingly, different ways of conceptualizing and structuring programming instruction at the precollege level are suggested. First, group means on four of five posttest measures were directionally higher for the limited-access group than the unlimited-access group. While it is clearly helpful to achievement to have sufficient access to computers, imposing reasonable limitations on computer access may encourage students to give greater attention to program design and mental execution of code. This overall interpretation, however, was qualified by several ATI effects. Females performed better under limited access, while males showed the opposite pattern. One possible explanation concerns females' generally lower confidence and greater anxiety regarding computer interactions (Chen, 1986). Thus, a greater proportion of females than males may have found it more comfortable to work without a computer. Given that this effect occurred on only one dependent variable, however, its importance should be questioned. A stronger and more consistent ATI pattern was for low-ability students to perform better with unlimited access, and conversely for middle- and high-ability students. Low-ability students, it would seem, are less able to mentally simulate program execution and therefore become more dependent on immediate computer feedback and concrete contextual cues. Higher-ability students are better prepared to benefit from the added cognitive demands of limited access and to use the increased planning time effectively.

The significant flowcharting effects on posttest scores indicated that students performed better when not required to submit flowcharts. One explanation is that many did not adequately master flowcharting skills. In fact, on a follow-up survey 65% reported understanding it "somewhat" and 19% "not at all." Second, seemingly because of its pictorial orientation and special symbol system, flowcharting appeared to be regarded by many students as an entirely separate task, rather than as a programming aid. Perhaps, similar to how sentence diagramming in English is taught, flowcharting may be more beneficial if introduced after students have acquired a fundamental understanding of programming processes. It is also revealing that over 65% of the flowchart group reported creating the flowchart after having written the associated programming code. These negative experiences undoubtedly contributed to the flowchart group's rating of programming as "more frightening" compared to the no-flowchart group. Gender was not significantly related to either achievement or attitudes. It is noteworthy, however, that in this elective high school literacy course, taught by a female instructor, females outnumbered males by 2:1, a direct contrast with typical ratios (Lockheed, 1985). As Linn (1985) has noted, the main problem for female students has traditionally been lack of participation, not of ability, in programming classes.

The above results suggest three major considerations for the teaching of programming. First, unlimited computer access may be less important than is generally assumed, especially for middle- and high-achievers. Teachers might consider encouraging (or requiring) the latter groups to spend more time designing and mentally simulating procedures away from the computer. Second, if flowcharting is to be useful as a design aid, considerably more time and emphasis should be given to its instruction than is done typically. Poorly formulated flowcharts can only provide weak foundations for developing programs. Third, as other recent studies have suggested, males and females appear to have much the same potential and instructional needs for learning programming. Seemingly, students' increasing exposure to computers in early

grades and to positive female role models (as in the present study) will make female participation in programming classes less of a problem over time.

References

Bayman, P. (1983, August). The effects of instructional procedures on beginning programmers' mental models. Paper presented at the annual meeting of the American Psychological Association, Anaheim, CA.

Bayman, P. & Mayer, R.E. (1983). Diagnosis of beginning programmers' misconceptions of BASIC programming statements. Communications of the ACM, 26, 677-679.

Brook, J.B., & Duncan, K.D, (1982). An experimental study of flowcharts as an aid in identification of procedural faults. Ergonomics, 23, 387-399.

Chen, M. (1986). Gender and computers: The beneficial effects of experience on attitudes. Journal of Educational Computing Research, 2, 265-282.

Dalbey, J., Tournaire, F., & Linn, M. (1986). Making programming instruction cognitively demanding: An intervention study. Journal of Research in Science Teaching, 23, 427-436.

Glorfeld, L.G., & Palko, J. (1984). Flowcharts in data processing education: An empirical review. The Journal of Data Education, 20, 18-20.

Linn, M.C. (1985). Gender equity in computer learning environments. Computers and the Social Sciences, 1, 19-27.

Lockneed, M.E. (1985). Women, girls, and computers: A first look at the evidence. Sex Roles, 13, 229-245.

Mandinach, E.B., & Corno, L. (1985). Cognitive engagement variations among students of different ability level and sex in a computer problem solving game. Sex Roles, 13, 241-251.

Mayer, R.E. (1979). A psychology of learning BASIC. Communications of the ACM, 589-593.

Pea, R.D., & Kurland, D.M. (1983). On the cognitive prerequisites of learning computer programming (Tech. Rep. No. 16). New York: Bank Street College of Education, Center for Children and Technology

Schneiderman, B., Mayer, R.E., McKay, D., & Heller, P. (1982). Experimental investigations of the utility of detailed flowcharts in programming. Communications of the ACM, 20, 373-381.